**Simple Active Filter**

![Simple Active Filter Diagram](image1)

- Select the cutoff frequency $f_c$.
- Divide the two capacitors by $2\pi f_c$.
- Select standard capacitors with the same order of magnitude.
- Adjust resistors to maintain $f_c$ (i.e., $R = 10k\Omega \cdot x$).

**Bandpass Filters**

- High-pass filter
- Low-pass filter

- Select a convenient capacitance value for the two capacitors.
- Calculate the three resistor values for $x = 1/(2\pi f_c C)$.
- Resistors should be in the $5k\Omega$ to $5M\Omega$ range. If not, repeat with a different capacitance value.

**Band-Reject Filters**

- High-pass filter
- Low-pass filter

- Multiple Feedback Bandpass Filter

![Multiple Feedback Bandpass Filter Diagram](image2)
Digital-to-Analog Converters

**DAC Parameters**

- **Precision** is number of distinguishable DAC outputs.
- **Range** is maximum and minimum DAC output.
- **Resolution** is smallest distinguishable change in output.

\[
\text{Range (volts)} = \text{Precision (alternatives)} \cdot \text{Resolution (volts)}
\]

- **Accuracy** is (actual-ideal)/ideal.
- Two common encoding schemes:

\[
V_{\text{out}} = V_{\text{fs}} \left( \frac{b_7}{2} + \frac{b_6}{4} + \frac{b_5}{8} + \frac{b_4}{16} + \frac{b_3}{32} + \frac{b_2}{64} + \frac{b_1}{128} + \frac{b_0}{256} \right) + V_{\text{os}}
\]

**Three-Bit DAC Examples**

**DAC Errors: Sources and Solutions**

- Errors can be due to:
  - Incorrect resistor values
  - Drift in resistor values
  - White noise
  - Op amp errors
  - Interference from external fields

- Solutions:
  - Precision resistors w/low tolerances
  - Precision resistors w/good temperature coefficients
  - Reduce BW w/low pass filter, reduce temperature
  - Use more expensive devices w/low noise and low drift
  - Shielding, ground planes

**DAC Using a Summing Amplifier**
Three-Bit DAC with an R-2R Ladder

Variable-Offset and Gain Using 3-bit DACs

Twelve-Bit DAC with a DAC8043
**DAC Selection: Precision, Range, and Resolution**
- Affect quality of signal that can be generated.
- More bits means finer control over the waveform.
- Can be hard to specify a priori.

**DAC Selection: Channels, Configuration, and Speed**
- Usually more efficient to implement multiple channels using a signal DAC.
- Configuration: can have voltage or current outputs, internal or external references, etc.
- Speed specified in many ways: settling time, maximum output rate, gain/BW product, etc.

**DAC Selection: Power and Interface**
- Three power issues: type of power required, amount of power required, and need for low-power sleep mode.
- Three approaches for interfacing exist:

**DAC Selection: Package and Cost**
- Variety of packages exist:
- Cost includes direct cost of components, power supply requirements, manufacturing costs, labor in calibration, and software development costs.

**DAC Waveform Generation**

```
unsigned short wave(unsigned short t){
  float result,time;
  time = 2*pi*((float)t)/1000.0;
  // integer t in msec into floating point time in seconds
  result =2048.0+1000.0*cos(31.25*time)-500.0*sin(125.0*time);
  return (unsigned short) result;
}
```

```c
#define RATE 2000
#define OC5 0x20
unsigned short Time; // Inc every 1ms
void interrupt 13 TOC5handler(void){
  TFLG1 = OC5; // ack C5F
  TC5 = TC5+RATE; // Executed every 1 ms
  Time++;
  DACout(wave(Time));
}
```
Periodic Interrupt Used to Generate Waveform

```c
unsigned short I; // incremented every 1ms
const unsigned short wave[32] = {
    3048,2675,2472,2526,2957,2931,2597,2048,1499,1165,1139,1341,1570,1624,1421,
    1048,714,624,863,1341,1846,2165,2206,2048,1890,1931,2250,2755,3233,3472,3382};
#define RATE 2000
#define OC5 0x20
void interrupt 13 TOC5handler(void){
    TFLG1 = OC5; // ack CSF
    TC5 = TC5+RATE; // Executed every 1 ms
    if(++I==32) {I=0; DACout(wave[I]);
    } else {
        DACout(wave[I] + ((wave[I+1] - wave[I]) * (I-t[I])) / (t[I+1]-t[I]));
    }
}
```

Chris J. Myers (Lecture 18: Filters/DACs) ECE/CS 5780/6780: Embedded System Design 27 / 35

Generated Waveform Using Uneven-Time

```c
unsigned short I; // incremented every sample
const unsigned short wave[32] = {
    3048,2675,2472,2526,2957,2931,2597,2048,1499,1165,1139,1341,1570,1624,1421,
    1048,714,624,863,1341,1846,2165,2206,2048,1890,1931,2250,2755,3233,3472,3382};
const unsigned short dt[32] = { // 500 ns cycles
#define OC5 0x20
void interrupt 13 TOC5handler(void){
    TFLG1 = OC5; // ack CSF
    if(++I==32) {I=0;DACout(wave[I]);
    } else {
        DACout(wave[I] + ((wave[I+1] - wave[I]) * (I-t[I])) / (t[I+1]-t[I]));
    }
}
```

Chris J. Myers (Lecture 18: Filters/DACs) ECE/CS 5780/6780: Embedded System Design 35 / 35